Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных Технологий
институт
Информационные системы
кафедра

УТВЕРЖДАЮ
Зав. кафедрой ИС
_____ Виденин С. А.
подпись инициалы, фамилия
«_____» _____ 2016 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии

Разработка кроссплатформенного web–приложения с использованием JavaScript фреймворков

Руководитель _____ С.А. Виденин
подпись, дата

Выпускник _____ С.Б. Платицин
подпись, дата

Нормоконтролер _____ Ю. В. Шмагрис
подпись, дата

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Проблема отсутствия единой централизованной информационной системы контроля олимпиад, проходящих в рамках мероприятий Сибирского Федерального Университета, остается актуальной на данный момент времени.

Университет нуждается в максимально отлаженном процессе управления олимпиадными мероприятиями и, как следствие, инструменте, способном реализовывать необходимые для этого задачи.

Целью выпускной квалификационной работы является разработка информационной системы, направленной на олимпиадную деятельность Сибирского Федерального Университета.

Для достижения поставленной цели требуется решить следующие задачи:

- Проанализировать выбранную предметную область;
- Выявить существующую проблематику;
- Выработать методы и способы решения задач;
- Выбрать оптимально подходящие средства разработки;
- Спроектировать бизнес – логику приложения;
- Разработать бизнес – логику приложения;
- Спроектировать базу данных;
- Разработать интерфейс информационной системы;
- Протестировать программный продукт;
- Внедрить программный продукт;

ГЛАВА 1 ОБЩИЕ СВЕДЕНИЯ

1.1 Проблематика предметной области

В современном интернет–пространстве существует множество готовых сервисов, как отечественных, так и зарубежных, которые предназначены для проведения олимпиадных мероприятий в учебных заведениях. Однако, существует ряд причин, по которым использование данных готовых решений если и не затруднит работу оргкомитета, проводящего мероприятия, то, как минимум, не позволит эффективно использовать ресурсы этих сервисов.

Одна из главных причин, вследствие которой создание собственного ресурса становится актуальной проблемой для университета, является необходимость доступа организаторами олимпиад к расширенным возможностям контроля ресурса, а также, при необходимости, его полного изменения и добавления нового функционала на основе постоянно изменяющихся правил проведения олимпиад.

Также следует выделить возрастную проблематику, так как не все ресурсы нацелены на работу с учащимися как общеобразовательного, так и высшего образовательного уровней. В связи с этим возникает потребность в использовании либо нескольких ресурсов разом (что делает работу организаторов менее эффективной), либо в создании своего собственного ресурса.

Существует множество зарубежных ресурсов, направленных на работу с олимпиадными мероприятиями. Одним из наиболее популярных в данной сфере является ресурс Бэйлорского университета icpc.baylor.edu. Главным недостатком данной информационной системы является отсутствие поддержки русского языка, вследствие чего взаимодействие пользователей, желающих принять участие в предстоящих мероприятиях, и данным ресурсом будет весьма затруднительным. Для решения этой проблемы организаторам олимпиад необходимо выдавать подробные инструкции по

эксплуатации на русском языке. Кроме того, ресурс базируется на проведении командных олимпиад, что делает невозможным его использование в случае проведения некомандных олимпиад, где каждый участник выступает сам за себя.

Заглавная страница ресурса icpc.baylor.edu изображена на рисунке 1.

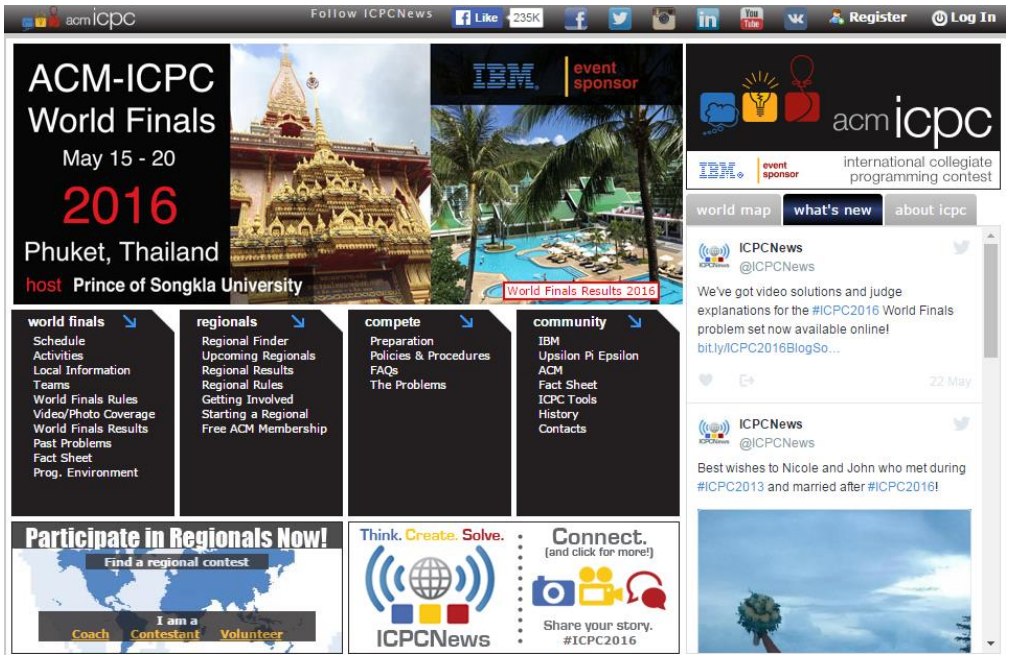

Рисунок 1 – Заглавная страница icpc.baylor.edu

Отечественный ресурс, разработанный Беляевым Сергеем Николаевичем, направлен, преимущественно, на школьную аудиторию. Учитывая данный факт, оргкомитет СФУ может столкнуться с проблемой, когда одновременно необходимо провести олимпиады, ориентированные на разные уровни учебных заведений, вследствие чего данный сервис удовлетворит потребности организаторов лишь частично.

Заглавная страница ресурса acmp.ru изображена на рисунке 2.

Исходя из всего вышесказанного, тема выпускной квалификационной работы является актуальной.

За счет введения университетом в эксплуатацию собственной информационной системы сократятся бизнес–процессы посредством их автоматизации, сократится время на выполнение этих процессов, а также

ускорится процесс организации олимпиадных мероприятий СФУ, в связи с чем повысится эффективность работы оргкомитета.

Информационная система позволит решить такие организационные задачи как: подача и обработка заявок на регистрацию, создание олимпиадных команд, выступающих под руководством тренеров, жеребьевка команд, объявление о проведении новых олимпиад и ведение журналов о прошедших мероприятиях.

Разрабатываемая информационная система будет реализована в виде web–приложения, предназначенного для автоматизации бизнес–процессов, проходящих внутри оргкомитета олимпиад Сибирского Федерального Университета.

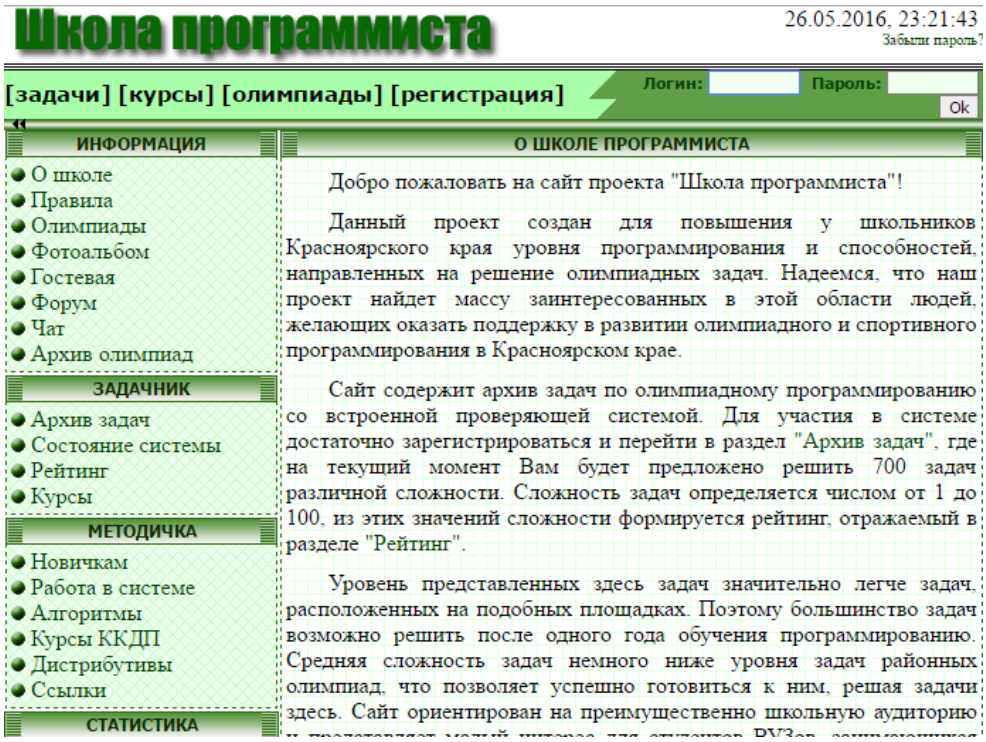

Рисунок 2 – Заглавная страница acmp.ru

Следует выделить основные бизнес–процессы, требующие автоматизации:

– регистрация пользователей на участие в олимпиадах;

– создание пользователями команд–участников олимпиад;

– выдача грамот и сертификатов участникам мероприятий;

– создание бейджиков для предоставления информации о его носителе;

– жеребьевка команд.

Кроме того, требуется определить основных участников информационной системы и их взаимодействие с системой. Разрабатываемая информационная система подразумевает четыре типа пользователей: «незарегистрированный пользователь», «участник», «тренер» и «администратор». Каждый из данных пользователей имеет ряд функциональных возможностей, предоставляемых системой, а также ряд ограничений, зависящих от уровня доступа. Диаграмма вариантов использования изображена на рисунке 3.

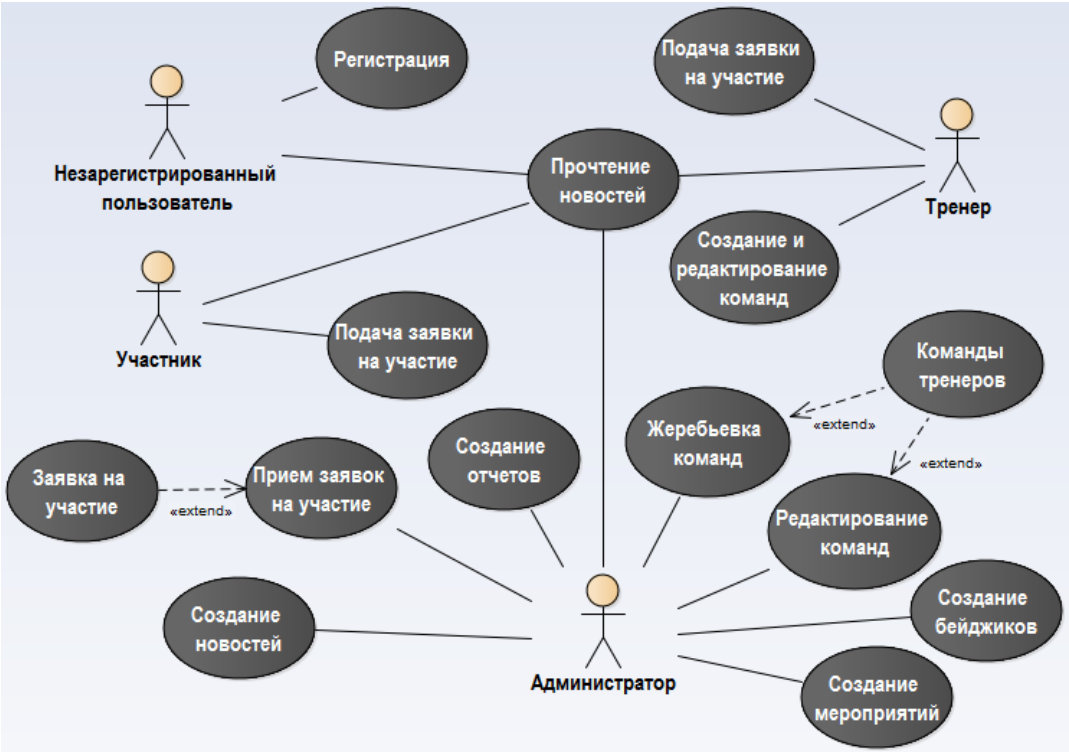

Рисунок 3 – Диаграмма вариантов использования

В соответствии с проанализированной проблематикой рассматриваемой предметной области, необходимо определить классификацию разрабатываемой информационной системы, провести

анализ проектируемой базы данных, а также вывести ряд требований, которым разрабатываемая информационная система должна удовлетворять.

1.2 Информационная система. Классификация разрабатываемой информационной системы

Информационная система (ИС) – это совокупность информационных, экономико–математических методов и моделей, технических, программных, технологических средств и специалистов, предназначенная для сбора, хранения, обработки и выдачи информации и принятия управленческих решений. Классические примеры информационных систем: 1С–Бухгалтерия, система SAP R3, справочно–правовая система «Консультант Плюс» и т.д.

Автоматизированная информационная система (АИС) – это информационная система, использующая электронно–вычислительную машину на этапах ввода, подготовки и вывода информации, т.е. она является развитием информационных систем, являющихся поиском с помощью прикладных программ.

Информационные системы можно классифицировать по различным признакам. В основу классификации положен ряд существенных различий, определяющих функциональные возможности и особенности построения современных систем. Наглядное представление классификаций ИС изображено на рисунке 4. Рассмотрим наиболее важные из них.

Рисунок 4 – Классификация информационных систем

По масштабу информационные системы делятся на однопользовательские, групповые и корпоративные.

Однопользовательские ИС – предназначены для личного использования на одном рабочем месте. На данный момент существует множество решений, направленных на автоматизацию рабочей деятельности частных лиц. В настоящее время на замену однопользовательским информационным системам пришли табличные процессоры, не имеющие какой–либо проблемной специализации, в первую очередь – Microsoft Excel.

Групповые ИС – предназначены для использования в отдельно взятых рабочих группах (малом предприятии, отделе и т.д.). В отличие от однопользовательских ИС, групповые системы, как правило, представляют собой специализированные клиентские решения для различных участников группы. Таковыми решениями могут являться такие системы как «Учебный отдел», «Деканат», «Кафедра», применительно к сфере учебного планирования.

Корпоративные ИС (КИС) – предназначены для автоматизации деятельности в крупных компаниях. КИС – это набор интегрированных

приложений комплексно поддерживающих все основные аспекты управленческой деятельности предприятий – планирование ресурсов для производства товаров (услуг), оперативное управление выполнение планов, все виды учета и анализ результатов хозяйственной деятельности.

Применительно к разрабатываемой информационной системе подходит классификация по масштабу «групповая», так как ее применение будет вестись в отдельно взятой рабочей группе, для ее различных участников.

Архитектура информационной системы – концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы. Существует три основных типа архитектур информационных систем:

– Архитектура «Файл–сервер»;
– Архитектура «Клиент–сервер»;
– Трехслойная архитектура.

Архитектура «Файл–сервер» – это архитектура, в которой исполняемые модули и данные размещаются в отдельных файлах операционной системы, доступ к которым осуществляется путем указания пути и использования файловых операций (открыть, считать, записать). Для хранения данных используется выделенный сервер (отдельный компьютер), который и является файловым сервером. Исполняемые модули хранятся либо на рабочих станциях, либо на файловом сервере.

Архитектура «Клиент–сервер» – это архитектура, в которой все вычисления или сетевая нагрузка распределены между серверами и клиентами. Клиент запрашивает те или иные сервисы в соответствии с определенным протоколом обмена данными. При этом, в отличие от ситуации с файловым сервером, нет необходимости в использовании прямых путей операционный системы: клиент их «не знает», ему «известны» лишь имя источника данных и другие специальные сведения, используемые для авторизации клиента на сервере. Сервер может находиться как на той же

вычислительной машине, так и удаленно, обрабатывая запросы со стороны клиента посредством интернет протоколов.

В системах подобного рода существует два основных «диалекта»: «толстый» и «тонкий» клиент.

В системах на основе «толстого» клиента, вся бизнес–логика реализована на стороне клиента, тогда как сервер представляет собой в чистом виде сервер баз данных, обеспечивающий исполнение только стандартизованных запросов на манипуляцию с данными (как правило, – чтение, запись, модификацию данных в таблицах реляционной базы данных).

Системы с «тонким» клиентом основаны на взаимодействии мощного сервера, выполняющего наибольшую часть бизнес–логики и клиента, не имеющего большой вычислительной мощности. Основное достоинство таких систем – относительная дешевизна клиентских станций.

Трехслойная архитектура – это архитектура, предполагающая наличие в себе трех составляющих: клиент, сервер приложений и сервер баз данных. С развитием интернет–технологий появилась разновидность трехслойной архитектуры на основании использования web–технологий. В этой разновидности в качестве сервера приложений играет web–сервер, а в качестве клиента используется стандартный web–браузер.

Применительно к разрабатываемой информационной системе подходит классификация по архитектуре «Клиент–сервер» на основе «толстого» клиента, так как это позволяет разрешить проблему использования информационной системы на различных клиентах с разными аппаратными платформами и операционными системами, а также позволит уменьшить нагрузку на серверную часть системы, т.к. большая часть вычислений будет приходится на сторону клиента.

Информационные системы по характеру использования информации можно разделить на информационно–поисковые и управляющие.

Информационно–поисковые системы, как правило, предоставляют доступ к хранимым данным исключительно в режиме чтения и используются

для поисков ответов на те или иные вопросы. Доступ к модификации данных ИС имеет администратор системы. В качестве примеров подобного рода систем можно привести поисковые ресурсы google.com и yandex.ru.

Управляющие системы – это системы, автоматизирующие деятельность, связанную с принятием решений. Действия конечных пользователей таких систем приводят к модификации информации, что не исключает возможности просто получать информацию. Системами такого типа могут являться системы бухгалтерского учета, системы планирования производственных ресурсов и т.д.

Применительно к разрабатываемой информационной системе подходит классификация по характеру использования информации «управляющая», т.к. действия пользователей так или иначе будут приводить к модификации информации.

Информационные системы делятся на ручные, автоматические и автоматизированные.

Ручные ИС – это информационные системы, в которых все операции производятся человеком без каких–либо технических средств переработки информации.

Автоматические ИС – это информационные системы, в которых все операции по переработке информации выполняются с помощью технических средств переработки информации без участия человека.

Автоматизированные ИС – это информационные системы, которые предполагают участие в процессе обработки информации и человека, и технические средства, причем главная роль в выполнении операций обработки данных отводится вычислительной машине. Именно этот класс систем соответствует современному представлению понятий «информационная система» и «автоматизированная система».

Применительно к разрабатываемой информационной системе подходит классификация по степени автоматизации «автоматизированная», т.к.

предполагается участие в процессе обработки информации, как человека, так и технических средств.

Резюмируя все вышесказанное, следует обобщить классификацию разрабатываемой ИС. Применительно к разрабатываемой информационной системе подходит классификация по масштабу – «групповая», по архитектуре – «Клиент–сервер» на основе «толстого» клиента, по характеру использования информации – «управляющая», по степени автоматизации – «автоматизированная». Классификация разрабатываемой ИС изображена на рисунке 5.

Рисунок 5 – Классификация разрабатываемой ИС

1.3 Базы данных. СУБД. Классификация проектируемой БД

База данных (БД) – это структурированная совокупность данных, хранимая в памяти вычислительной машины, которая характеризует состав объектов предметной области, их свойства и взаимосвязи. Кроме данных, БД содержит в себе программные методы и средства, позволяющие оперировать данными, содержащимися в базе.

Базы данных обладают признаками, отличающими их от других объектов, подходящих под понятие «совокупность хранимых данных».

Базы данных имеют следующие отличительные признаки:

– БД хранятся и обрабатываются исключительно посредством ресурсов вычислительной системы;

– для эффективной обработки и поиска в вычислительной системе все данные, хранящиеся в БД, структурированы;

– в БД встраивается логическая структура данных в соответствии с моделью, заложенной при разработке базы.

Таким образом, базами данных нельзя назвать такие совокупности хранимых данных как картотеки, архивы документов и т.п.

Все базы данных можно классифицировать по трем главным свойствам:

– по характеру хранимой информации;

– по способу хранения данных;

– по структуре организации данных.

Наглядное представление классификаций БД изображено на рисунке 6. Рассмотрим наиболее важные из них.

Рисунок 6 – Классификация БД

По характеру хранимой информации БД делятся на фактографические и документальные.

Фактографические базы данных содержат в себе сведения об объектах, объединенных какой–либо предметной областью. Такими объектами могут быть библиотечные книги, товары, а их сведениями – год издания книги и цена товара.

Документальные базы данных содержат в себе описания каких–либо документов. Как правило, в зависимости от содержания описания различают три типа документальных БД:

– имеющие только библиографическое описание;

– имеющие библиографическое описание и ключевые слова;

– имеющие библиографическое описание, ключевые слова и аннотацию.

Применительно к разрабатываемой информационной системе подходит классификация проектируемой базы данных «документальная», т.к. данные, содержащиеся в базе, будут иметь в себе описания каких–либо документов.

По способу хранения данных БД делятся на централизованные и распределенные.

Централизованные базы данных хранятся на одной вычислительной машине, которая в свою очередь может быть автономным персональным компьютером, либо сервером в сети, доступ к которому может быть многопользовательским, либо параллельным, т.е. независимым.

Распределенные базы данных распределены между компьютерами, находящимися в одной сети, как локальной, так и сети интернет. Такой подход к хранению данных используется в локальных и глобальных компьютерных сетях.

Применительно к разрабатываемой информационной системе подходит классификация проектируемой базы данных «централизованная», т.к. ее хранение будет осуществляться на одной вычислительной машине.

По структуре организации данных БД делятся на реляционные, нереляционные, иерархические и сетевые.

Реляционные БД представляют собой данные, имеющие табличную форму организации.

Нереляционные документно–ориентированные БД или NoSQL базы данных используют, в отличие от реляционных БД, не регламентированную структуру, т.е. в отдельной строке или документе можно добавить произвольное поле без предварительного изменения структуры всей таблицы. Характерными чертами NoSQL баз данных является использование многопроцессорности, что позволяет повысить производительность, а также масштабируемость.

Иерархические БД реализованы в виде древовидной (иерархической) структуры, состоящей из объектов различных уровней, где у каждого объекта может иметься одна запись–потомок.

Сетевые БД являются, в свою очередь, расширением иерархического подхода реализации, но, в отличие от иерархических баз данных, запись–потомок может иметь любое число записей–предков.

Применительно к разрабатываемой информационной системе подходит классификация проектируемой базы данных «нереляционная», т.к. именно нереляционная база данных позволит повысить производительность разрабатываемой ИС, а также позволит при необходимости легко производить масштабирование.

Система управления базами данных (СУБД) – это комплекс программных средств, предназначенных для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и визуализации информации.

Существует два типа СУБД – настольные и серверные.

Настольные СУБД представляют с собой клиентские приложения, обеспечивающие чтение, запись и обработку данных БД. Взаимодействие с

данными осуществляется посредством файловых сервисов операционной системы.

Серверные СУБД обеспечивают обработку данных, находящихся в общедоступных хранилищах, как правило, на сервере данных, для нескольких пользователей одновременно. Серверные СУБД реализуют передачу запросов серверу БД и получение результатов этих запросов или кодов ошибок клиенту. СУБД данного типа основаны на архитектуре «клиент–сервер».

Применительно к разрабатываемой информационной системе будет использована серверная СУБД, так как необходимо обеспечение обработки данных, находящихся на сервере.

Резюмируя все вышесказанное, следует обобщить классификацию проектируемой базы данных. Применительно к проектируемой базе данных подходит классификация по характеру хранимой информации – «документальная», по способу хранения данных – «централизованная» по структуре организации данных – «нереляционная». Классификация проектируемой базы данных изображена на рисунке 6.

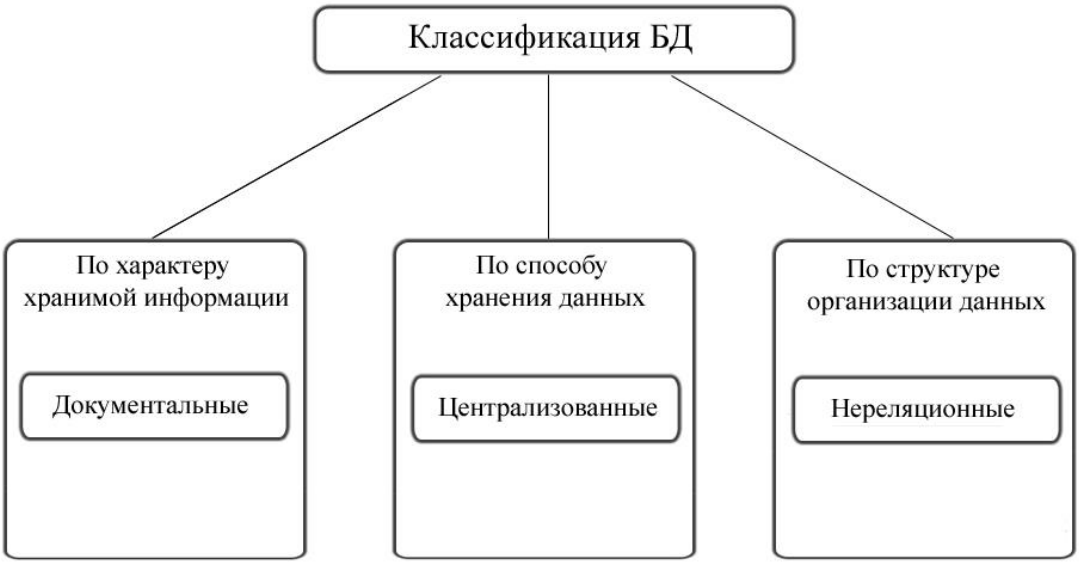

Рисунок 6 – Классификация проектируемой базы данных

1.4 Требования к разрабатываемой ИС

Проведя анализ предметной области, а также выявив классификацию разрабатываемой системы и проектируемой для нее базы данных, необходимо разработать ряд требований, которыми должна отвечать разрабатываемая в рамках выпускной квалификационной работы информационная система.

Разрабатываемое web–приложение должно отвечать следующим техническим требованиям:

– быстродействие (высокая скорость при вводе, поиске и обработке информации);

– отказоустойчивость при больших нагрузках;

– централизация данных в единой базе;

– кроссплатформенность (система должна одинаково хорошо работать на различных клиентах с разными аппаратными платформами и операционными системами);

– модульность (для более быстрого процесса разработки новых функций ресурса с помощью написания и внедрения новых модулей системы).

Кроме того, приложение должно отвечать следующим требованиям к безопасности:

– идентификация и аутентификация пользователей;

– защита пользовательской информации путем ее шифрования;

– разделение доступа к функциям в зависимости от прав пользователя.

Также, система должна иметь интуитивно понятный графический интерфейс. Взаимодействие с интерфейсом должно осуществляться с помощью манипуляторов типа «мышь», а ввод данных с помощью клавиатуры.

ГЛАВА 2 РАЗРАБОТКА КРОСПЛАТФОРМЕННОГО WEB–ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ JAVASCRIPT ФРЕЙМВОРКОВ

2.1 Описание бизнес–процессов

Согласно проведенному исследованию предметной области, были выведены следующие основные бизнес–процессы, которые необходимо автоматизировать в разрабатываемой информационной системе:

– регистрация пользователей на участие в олимпиадах;

– создание грамот и сертификатов участникам мероприятий;

– создание бейджиков для предоставления информации о его носителе;

– жеребьевка команд.

Необходимо описать модели «как есть» и «как надо» для каждого из приведенных выше бизнес–процессов.

Модель «как есть» (англ. «as is») каждого бизнес–процесса описывает и отражает ход процесса, действия, роли, движение документов, а также точки возможной оптимизации. Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых бизнес–процессов и насколько глубоким изменениям подвергнется существующая структура организации бизнеса.

Модель «как надо» (англ. «to be») или «как должно быть» представляет собой переработанную версию модели «как есть» с учетом уже проанализированных и автоматизированных бизнес–процессов. Найденные недостатки в модели «как есть» исправляются при создании модели «как надо».

Следует определить категории пользователей, имеющие непосредственное участие в информационной системе, а так же их роль и доступные функции, которые им предоставляет ИС.

В системе имеется четыре категории пользователей: гость сайта, т.е. незарегистрированный в системе пользователь, участник олимпиады, тренер и администратор. Каждый из них работает с системой как отдельный пользователь, получая доступ лишь к своей части информации и функциональным возможностям.

Гость сайта имеет ограниченный доступ к системе и минимальный набор функциональных возможностей. В основные возможности, предоставляемые ИС незарегистрированному пользователю, входит:

– возможность регистрации и авторизации в информационной системе;

– ознакомление с новостными статьями.

Диаграмма прецедентов для незарегистрированного пользователя изображена на рисунке 8.

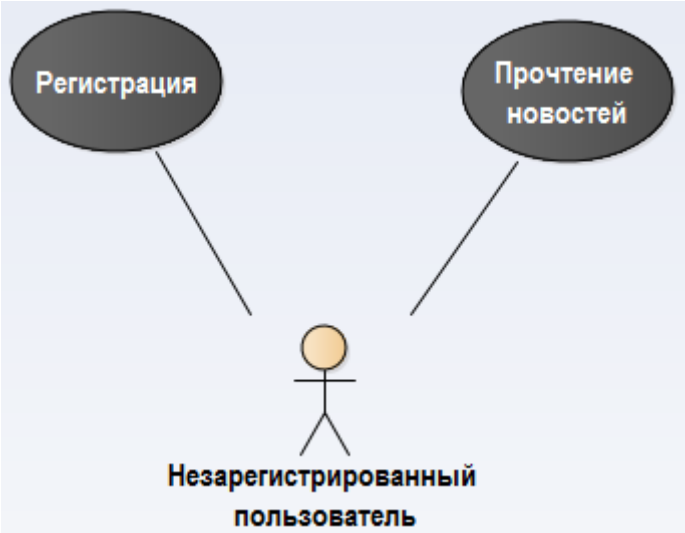

Рисунок 8 – Диаграмма прецедентов незарегистрированного пользователя

Участник олимпиад помимо функций, доступных незарегистрированному пользователю, имеет дополнительный набор функциональных возможностей:

– редактирование личной информации в профиле пользователя;

– возможность подавать заявку на участие в олимпиаде.

Диаграмма прецедентов для участника олимпиад изображена на рисунке 9.

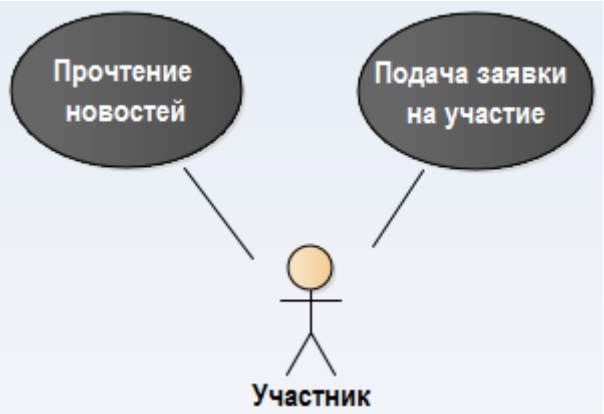

Рисунок 9 – Диаграмма прецедентов участника олимпиад

Тренеру команд доступен следующий функционал:

– редактирование личной информации в профиле пользователя;

– возможность создавать команды и добавлять в них участников.

Диаграмма прецедентов для тренера команд изображена на рисунке 10.

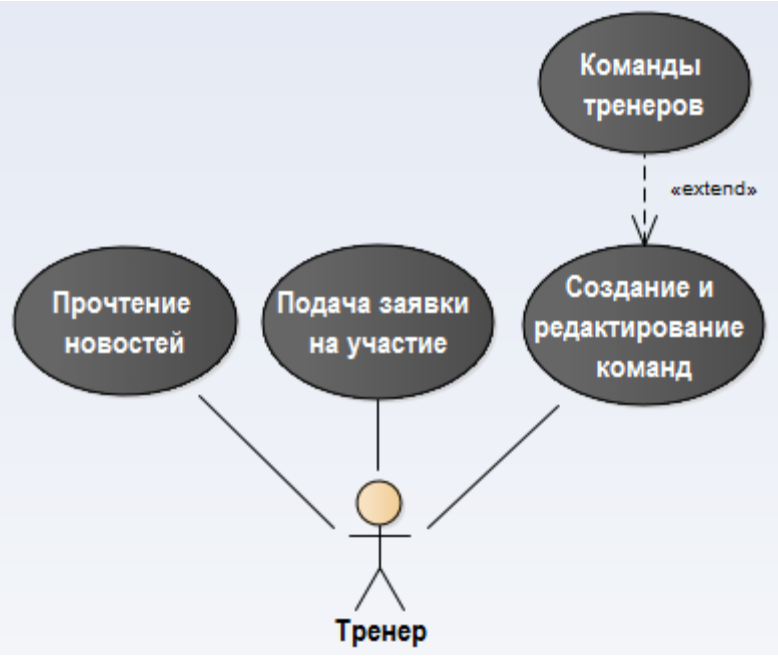

Рисунок 10 – Диаграмма прецедентов тренера команд

Администратор имеет наибольший спектр возможностей по работе с информационной системой. Администратору доступны следующий перечень функций:

– редактирование команд, создаваемых тренерами;
– создание новостей;
– создание олимпиадных мероприятий;
– прием заявок как от отдельно взятых участников, так и от команд на участие в олимпиаде
– проведение жеребьевки команд.

Диаграмма прецедентов администратора изображена на рисунке 11.

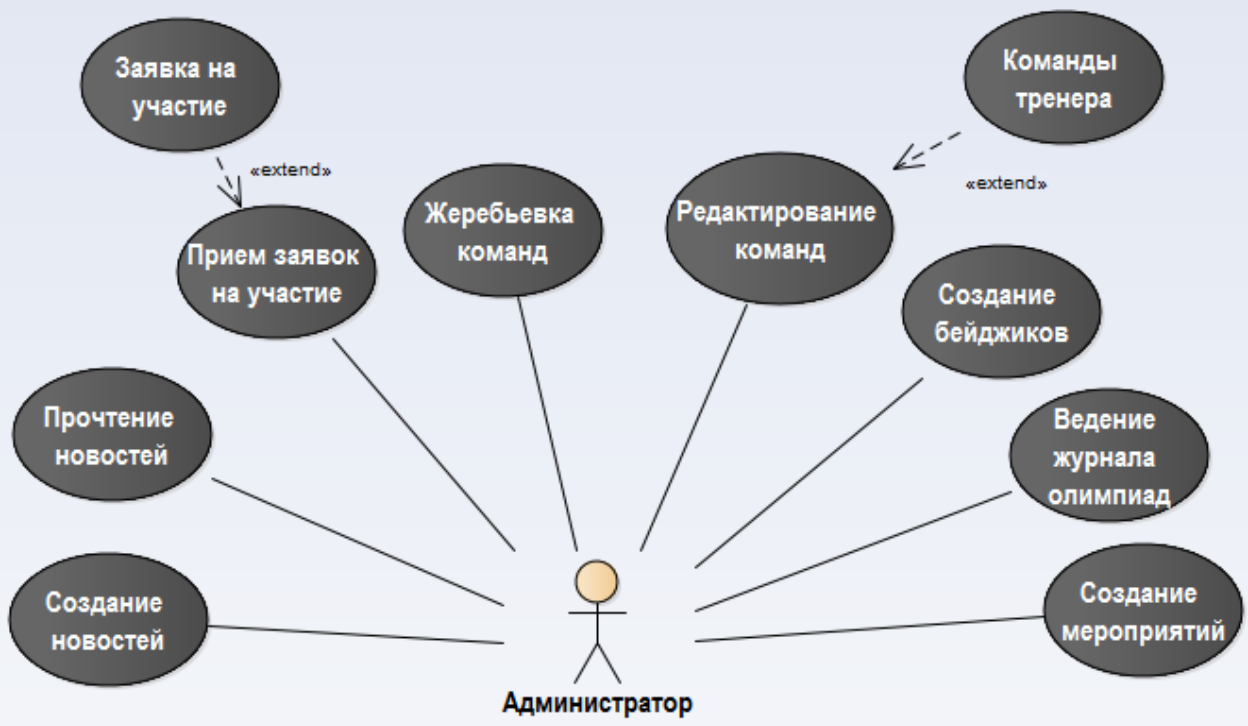

Рисунок 11 – Диаграмма прецедентов администратора

2.1.1 Модель регистрации участников и команд

Модель «как есть»

Неавтоматизированный процесс регистрации участников и команд осуществляется путем отправки заполненного бланка об участии, в котором

содержатся основные данные об участнике или, в случае регистрации команды – участниках. После получения бланка оргкомитетом олимпиады, все данные вносятся в журнал. Для повторного участия в последующих олимпиадных мероприятиях требуется повторение вышеуказанной процедуры регистрации. Диаграмма деятельности неавтоматизированного процесса регистрации участников и команд изображена на рисунке 12.

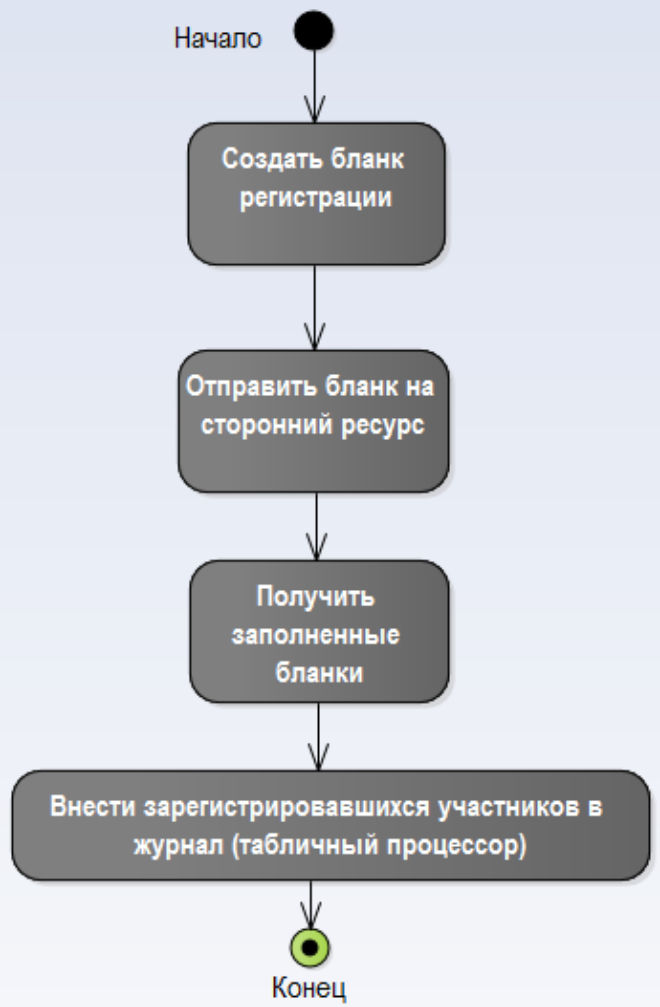

Рисунок 12 – Диаграмма деятельности регистрации участников и команд

Входящим событием данного бизнес–процесса является потребность команды или отдельно взятого участника в подаче заявления на регистрацию участия в олимпиадном мероприятии.

Границами данного бизнес–процесса являются создание, просмотр и редактирование заявок на участие.

Основными участниками являются оргкомитет олимпиадного мероприятия, тренера команд или отдельно взятые участники.

Входящими документами являются заявления на участие в олимпиаде, отправляемые организатору олимпиадных мероприятий в виде электронного письма с приложенным к нему текстовым документом.

Исходящим событием будет являться подтверждение об участии от организатора, а также занесение команд или отдельных участников в список участников конкретно взятой олимпиады.

SADT диаграмма данного бизнес–процесса изображена на рисунке 13.

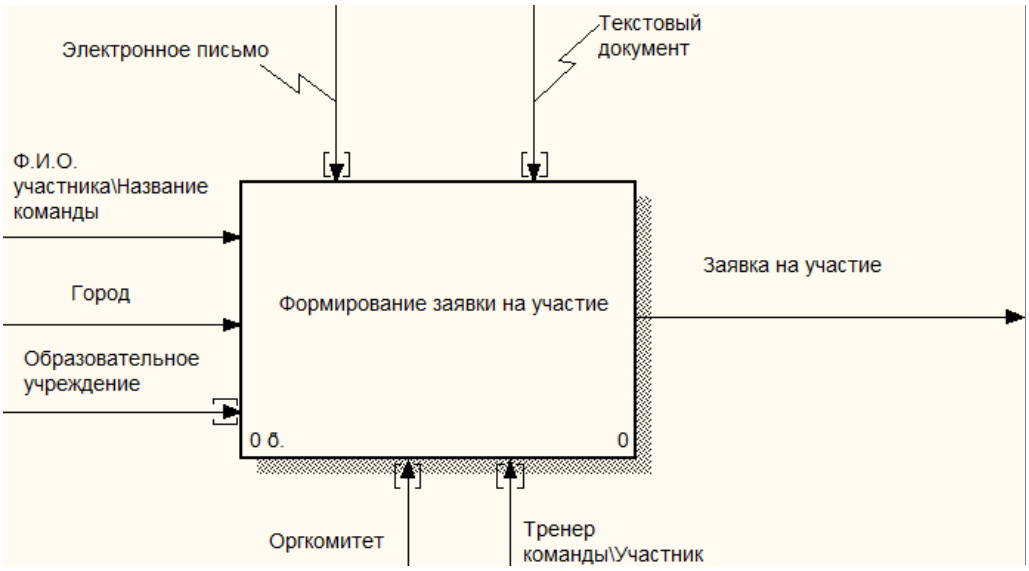

Рисунок 13 – SADT диаграмма регистрации участников

Модель «как надо»

Автоматизированный процесс регистрации команд или отдельно взятых участников исключает повторную отправку данных оргкомитету олимпиад, т.к. при первой регистрации пользователей в системе все данные заносятся в базу данных и могут быть использованы в последующих мероприятиях. Диаграмма деятельности автоматизированного процесса регистрации участников и команд изображена на рисунке 14.

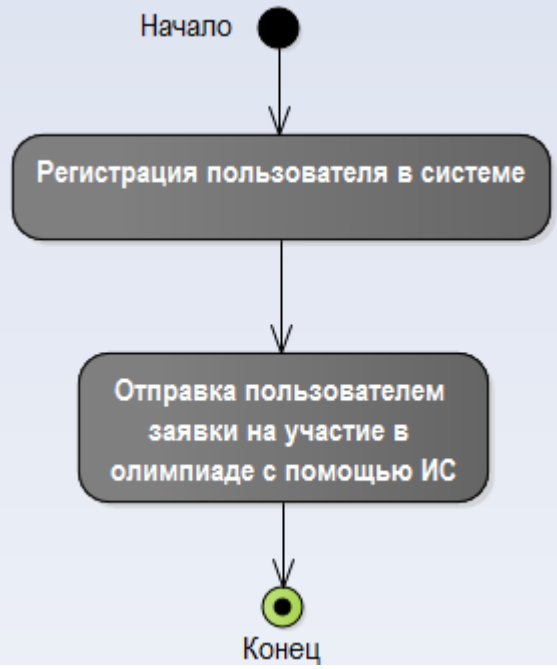

Рисунок 14 – Диаграмма деятельности автоматизированного процесса регистрации участников и команд

2.1.2 Модель создания грамот и сертификатов

Модель «как есть»

Неавтоматизированный процесс создания грамот и сертификатов предполагает ручное заполнение документов при участии одного из участников оргкомитета олимпиад. При этом используются данные, вручную внесенные в журнал участников при их регистрации в мероприятии. Диаграмма деятельности неавтоматизированного процесса создания грамот и сертификатов изображена на рисунке 15.

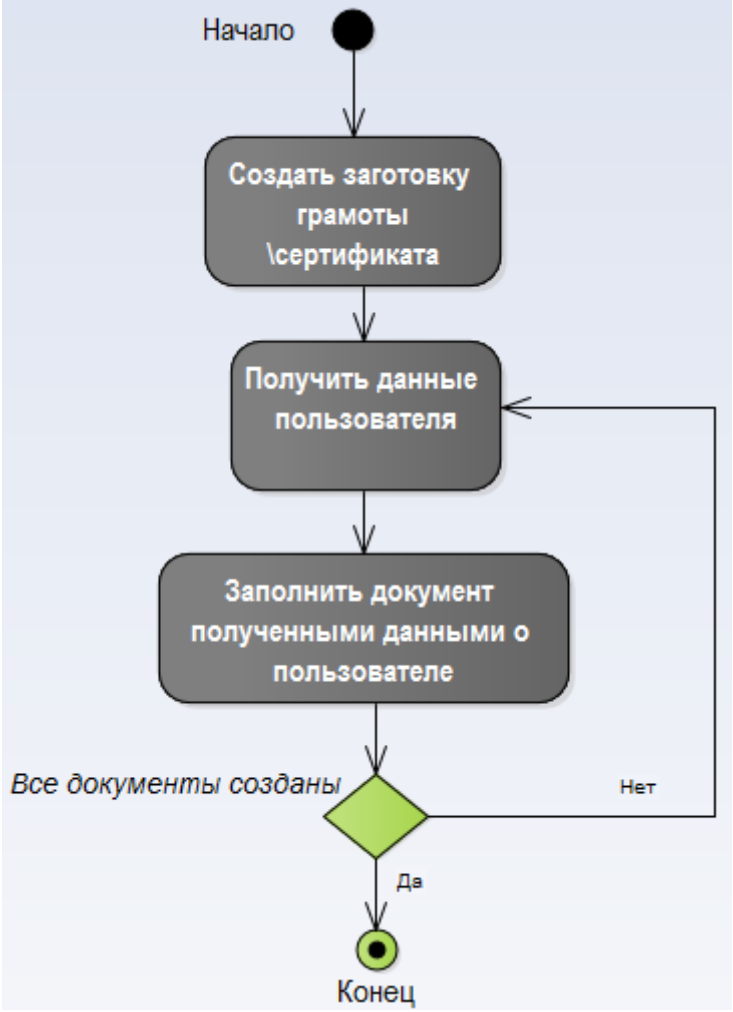

Рисунок 15 – Диаграмма деятельности создания грамот и сертификатов

Входящим событием данного бизнес–процесса является потребность в выдаче грамот и сертификатов участникам принимавшим участие в олимпиадном мероприятии.

Границами данного бизнес–процесса являются создание и передача грамот и сертификатов конечным владельцам.

Основными участниками являются организатор олимпиадного мероприятия, тренера команд или отдельно взятые участники.

Входящими документами являются данные, полученные в заявке на участие в олимпиадном мероприятии.

Исходящим событием является создание на основе полученных личных данных участников грамот и сертификатов, а также их передача конечным владельцам.

SADT диаграмма данного бизнес–процесса изображена на рисунке 16.

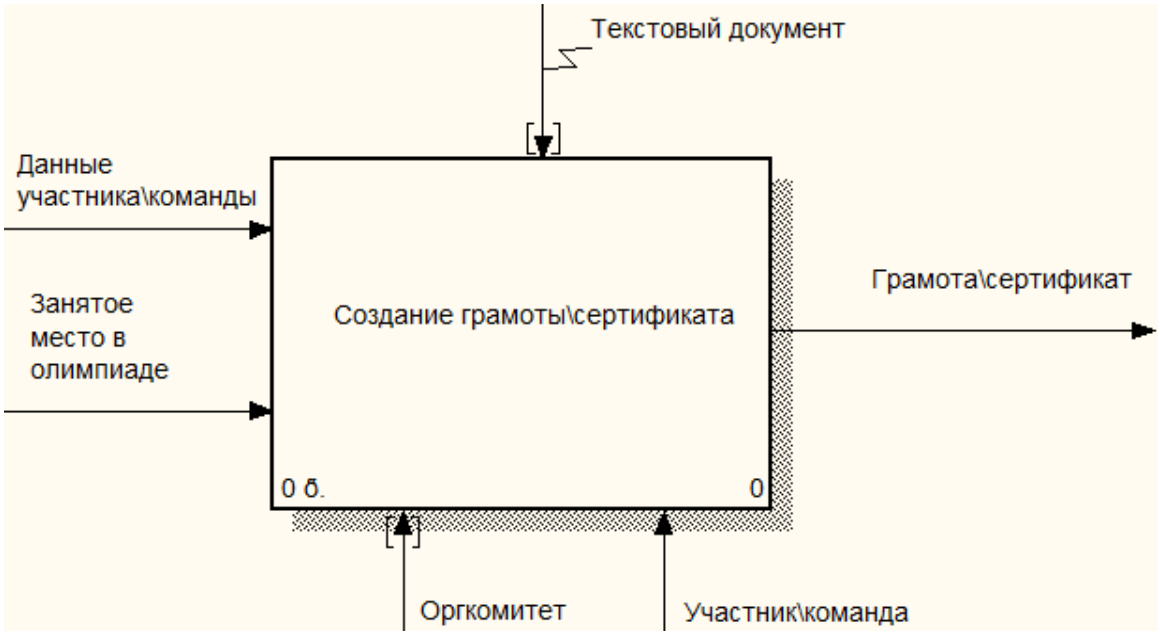

Рисунок 16 – SADT диаграмма создания грамот и сертификатов

Модель «как надо»

Автоматизированный процесс создания грамот и сертификатов исключает ручное заполнение документов, т.к. информационная система предполагает наличие специальных функциональных возможностей, способных выполнять данное действие нажатием одной кнопки. Пользователю системы необходимо выбрать участника или команду, нажать кнопку создания грамот и информационная система сделает все сама, используя заранее приготовленный шаблон грамот или сертификатов, добавляя в каждый из них по отдельности данные каждого участника, хранящиеся в базе данных системы. Диаграмма деятельности автоматизированного процесса создания грамот и сертификатов изображена на рисунке 17.

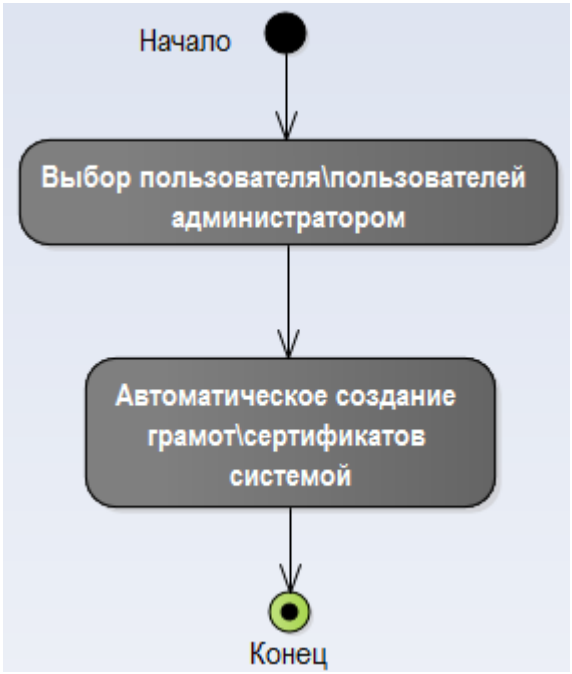

Рисунок 17 – Диаграмма деятельности автоматизированного процесса создания грамот и сертификатов

2.1.3 Модель создания бейджиков

Модель «как есть»

Неавтоматизированный процесс создания бейджиков предполагает ручное заполнение каждого из них, с использованием данных, полученных при регистрации участников. Диаграмма деятельности неавтоматизированного процесса создания бейджиков для участников олимпиад изображена на рисунке 18.

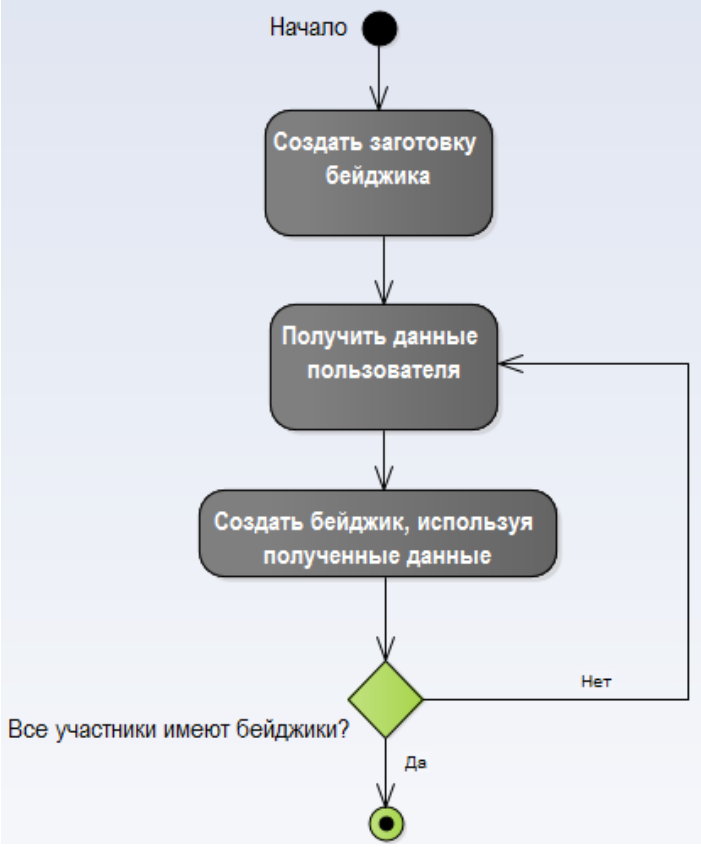

Рисунок 18 – Диаграмма деятельности создания бейджиков

Входящим событием данного бизнес–процесса является необходимость выдачи каждому участнику олимпиады собственного уникального бейджика для идентификации его носителя.

Границами данного бизнес–процесса являются создание и передача бейджиков конечным владельцам.

Основными участниками является оргкомитет олимпиадного мероприятия и отдельно взятые участники.

Входящими документами являются данные, полученные в заявке на участие в олимпиадном мероприятии.

Исходящим событием является создание, на основе полученных личных данных участников, бейджиков, а также их передача конечным владельцам.

SADT диаграмма данного бизнес–процесса изображена на рисунке 19.

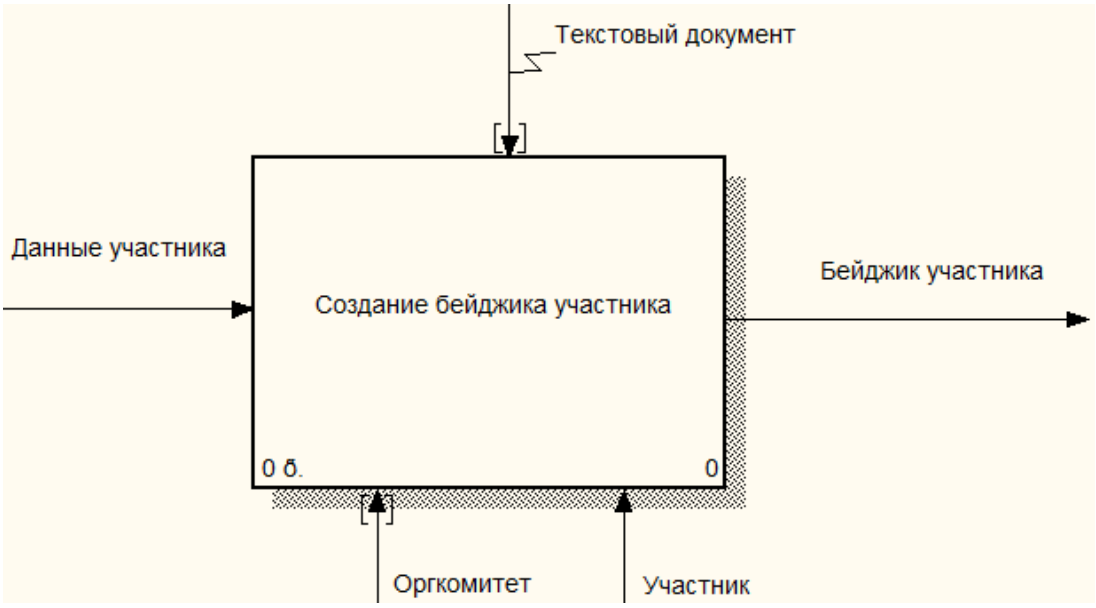

Рисунок 19 – SADT диаграмма создания беджиков

Модель «как надо»

Автоматизированный процесс создания бейджиков для участников олимпиад исключает их ручное заполнение и предполагает выполнение данного действия с использованием функциональных возможностей, встроенных в систему. Диаграмма деятельности автоматизированного процесса создания бейджиков изображена на рисунке 20.

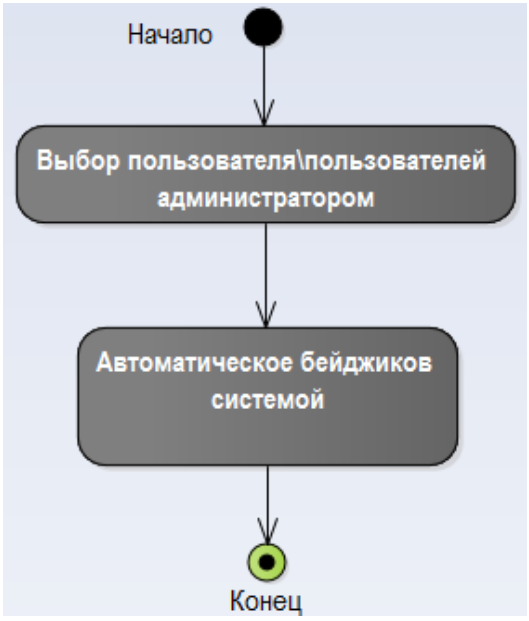

Рисунок 20 – Диаграмма деятельности автоматизированного процесса создания бейджиков

2.1.4 Модель жеребьевки

Модель «как есть»

Жеребьевка команд необходима для распределения участников по аудиториям случайным образом. Неавтоматизированный процесс жеребьевки выполняется оргкомитетом перед каждым этапом олимпиады. Диаграмма деятельности неавтоматизированного процесса жеребьевки изображена на рисунке 21.

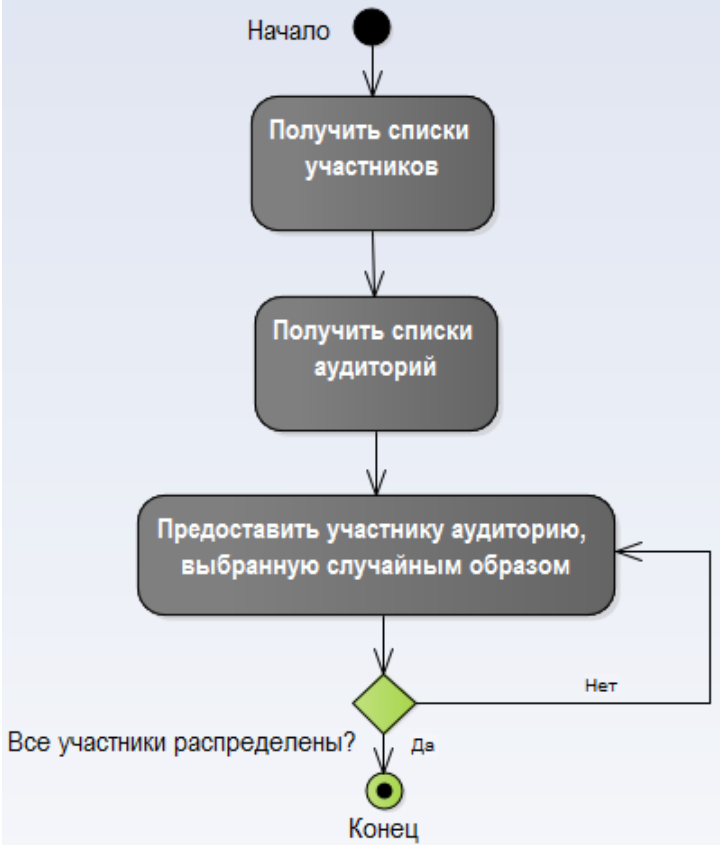

Рисунок 21 – Диаграмма деятельности процесса жеребьевки

Входящим событием данного бизнес–процесса является необходимость случайного распределения каждого участника по выбранным случайным образом аудиториям.

Границами бизнес–процесса является появление участников на мероприятии и их последующее распределение по аудиториям.

Основными участниками являются оргкомитет олимпиадного мероприятия и участники олимпиадного соревнования.

Входящим документом является список аудиторий.

Исходящим событием является распределение участников по аудиториям.

SADT диаграмма данного бизнес–процесса изображена на рисунке 22.

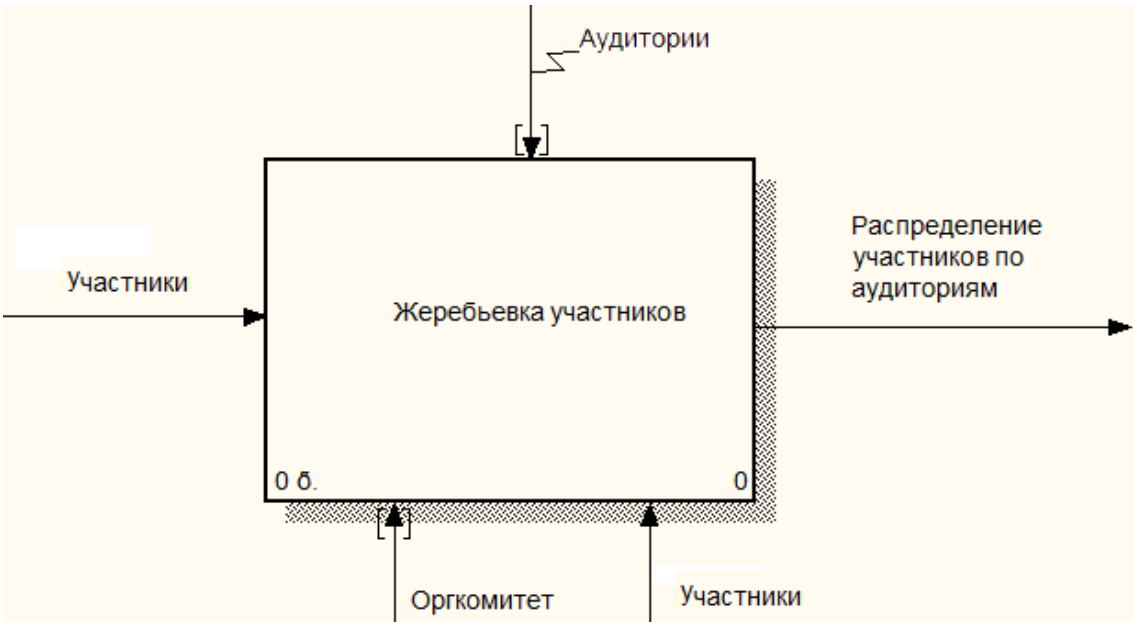

Рисунок 22 – SADT диаграмма жеребьевки

Модель «как надо»

Автоматизация данной модели предполагает наличие в системе алгоритма, позволяющего в автоматическом режиме, по команде пользователя системы распределить участников по аудиториям. Диаграмма деятельности автоматизированного процесса жеребьевки изображена на рисунке 23.

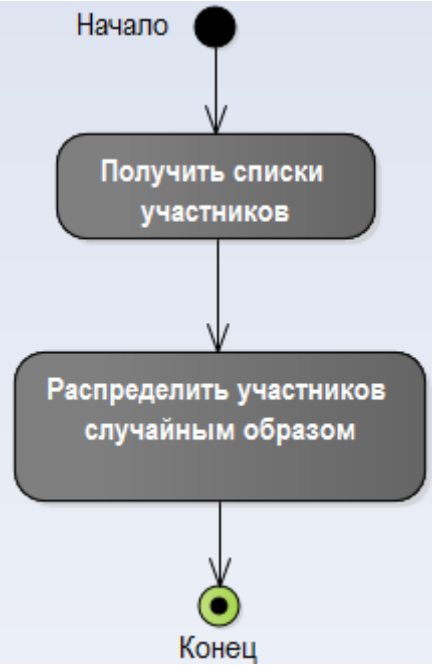

Рисунок 23 – Диаграмма деятельности автоматизированного процесса жеребьевки

2.2 Архитектура разрабатываемой информационной системы

Архитектура разрабатываемой информационной системы была выбрана с учетом требований, выдвинутых вследствие анализа предметной области. Следует провести краткий обзор инструментальных средств разработки, необходимых для реализации данной архитектуры.

Для достижения целей, поставленных в рамках выпускной квалификационной работы, был использован стек технологий MEAN. Стек (англ. stack) – это набор программного обеспечения, используемого для разработки. MEAN представляет собой слово, сложенное из первых букв используемых в нем фреймворков. Фреймворк (англ. framework) – это «каркас» будущей информационной системы, состоящий из множества библиотек. В данный стек входят такие программные средства разработки, как фреймворк AngularJS в качестве клиент–составляющей, платформа Node.js в качестве сервер–составляющей, фреймворк Express, необходимый в разработке на сервер–стороне, а также MongoDB как средство управления базами данных. Логотип MEAN изображен на рисунке 24.

Рассмотрим подробнее программное обеспечение, входящее в данный стек.

Рисунок 24 – Логотип стека MEAN

AngularJS

AngularJS – это фреймворк с открытым исходным кодом, использующий шаблон проектирования MVW (Model–View–Whatever) и предназначенный для разработки одностраничных веб–приложений с использованием языка программирования JavaScript. Модель MVW означает Model–View–Whatever (Модель–Вид–Все что угодно). Можно выбрать одну из двух моделей: MVC (Model–View–Controller) и MVVM (Model–View–ViewModel). Данный фреймворк нацелен на работу с языком разметки HTML, содержащим дополнительные пользовательские атрибуты – директивы. Директивы позволяют разработчику самому дописать поведение тех или иных HTML элементов. Основная особенность AngularJS – возможность изменять содержимое страницы без ее перезагрузки. Логотип фреймворка AngularJS изображен на рисунке 25.

Рисунок 25 – Логотип фреймворка AngularJS

Node.js

Node или Node.js — это кроссплатформенная программная платформа с открытым исходным кодом, основанная на JavaScript движке V8

(транслирующем JavaScript в машинный код) и предназначенная для разработки сервер–составляющих веб–приложений. JavaScript, язык программирования на котором базируется Node.js, является узкоспециализированным, однако платформа Node.js превращает его из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++)

На данный момент для платформы Node.js написано большое количество сторонних модулей, способных расширить возможности приложений на базе Node.js. Логотип Node.js изображен на рисунке 26.

Рисунок 26 – Логотип платформы Node.js

Express

Express.js – это фреймворк для Node.js, спроектированный для создания одностраничных, мультистраничных и гибридных веб–приложений. Де-факто является стандартным серверным фреймворком для Node.js. Логотип фреймворка изображен на рисунке 27.

Рисунок 27 – Логотип фреймворка Express

MongoDB

MongoDB – документо–ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц.

В отличие от других СУБД, MongoDB имеет ряд существенных отличий. Данные в MongoDB хранятся в формате BSON (Binary JSON), что позволяет выполнять их обработку и поиск на порядок быстрее. Вместо привычных SQL таблиц, MongoDB использует так называемые «коллекции», которые могут содержать в себе объекты, имеющие различную структуру и набор свойств. Кроме того, в базах MongoDB отсутствует понятие «первичный ключ»; вместо этого используется уникальный идентификатор (_id), который генерируется СУБД автоматически. Логотип СУБД MongoDB изображен на рисунке 28.

Рисунок 28 – Логотип СУБД MongoDB

2.2.1 Уровень данных

В проектировании баз данных разрабатываемого веб–приложения был использован документоориентированный подход вместо традиционного реляционного, поэтому здесь отсутствует понятие «таблица». Вместо него используется такое понятие как «коллекция». База данных разрабатываемого веб–приложения содержит 5 коллекций, содержащих данные об участниках олимпиад, тренерах, командах и новостях. На рисунке 28 изображен пример коллекции «Olympiad_CoachLogins», отвечающей за хранение в разрабатываемом веб–приложении личных данных тренеров (т.е. логин, пароль, ФИО и т.д.).

```
//db.coach_logins
var Olympiad_CoachLogins = new mongoose.Schema({
    login: {type: String},
    email: {type: String},
    password: {type: String},
    hash: {type: String},
    coachFirstName: {type: String},
    coachSecondName: {type: String},
    coachCity: {type: String},
    coachInstitution: {type: String},
    coachAvatar: {type: String}
});
```

Рисунок 29 – Коллекция «Olympiad_CoachLogins»

На рисунке 30 представлена схема базы данных, спроектированной в ходе разработки веб–приложения.

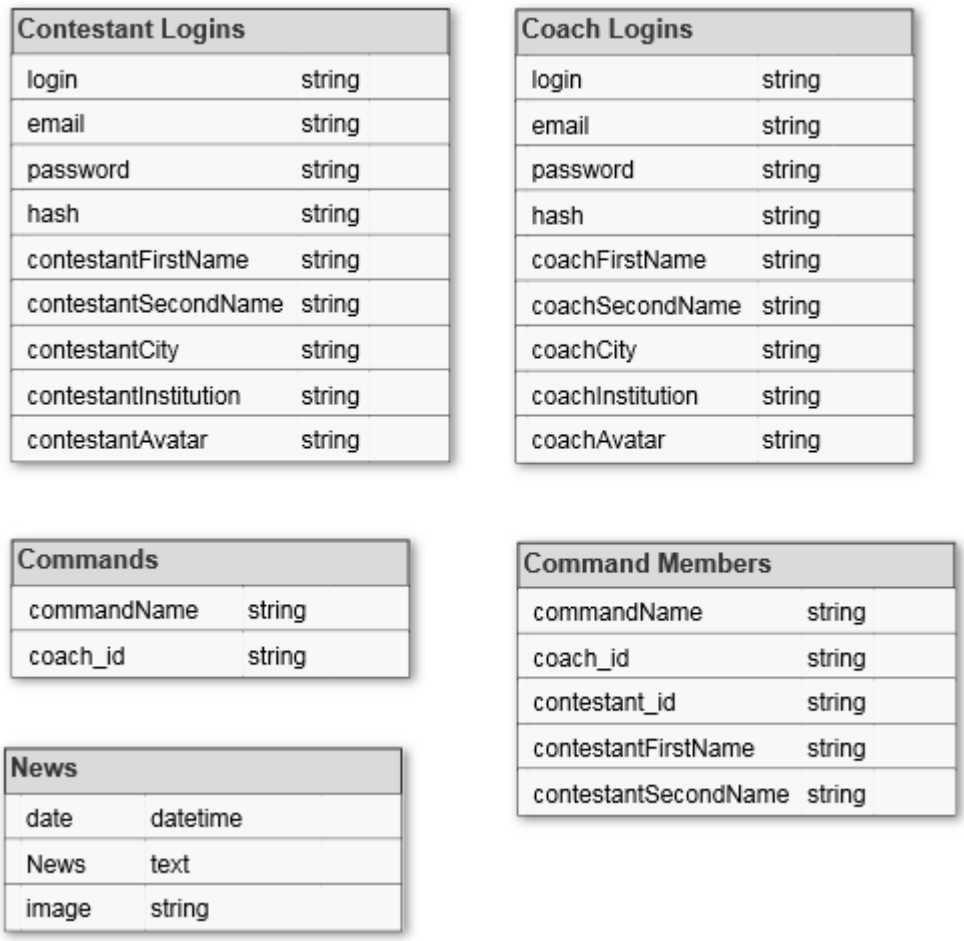

Рисунок 30 – Схема спроектированной базы данных

Формат данных BSON и JSON

Используемая в ходе разработки веб–приложения СУБД MongoDB хранит все полученные данные в формате BSON. BSON (англ. Binary JavaScript Object Notation) – это формат электронного обмена цифровыми данными, основанный на JavaScript, бинарная форма представления простых структур данных и ассоциативных массивов.

Несмотря на то, что хранимые в BSON формате данные занимают намного больше места на носителях информации, такой способ хранения позволяет выполнять обработку и поиск данных на порядок быстрее, чем в обычных реляционных базах данных.

После того, как сервером баз данных был получен определенный запрос от сервера приложения и сформирован ответ, все данные, содержащиеся в ответе, конвертируются из формата BSON в JSON формат и отправляются серверу приложения. Пример представления данных в BSON и JSON форматах изображен на рисунке 29.

Необходимо пояснить устройство данных в формате BSON. На представленном примере, каждая строчка данных в формате BSON имеет свое значение. В данном случае первая строка отвечает за размер документа, вторая строка отвечает за тип хранящейся строки, третья строка содержит в себе имя поля, четвертая строка содержит непосредственно содержимое поля (в данном случае слово «world»), пятая строка указывает на конец объекта.

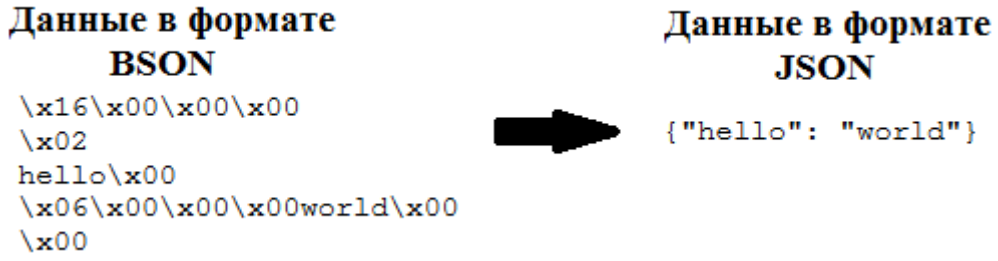

Рисунок 29 – Пример представления данных в BSON и JSON форматах

Полученный после обработки MongoDB документ в формате JSON легко обрабатывается сервером приложения. В свою очередь JSON формат

представляет собой текстовый формат обмена данными, основанный на языке программирования JavaScript и обычно используемый именно с этим языком. В отличие от BSON формата, JSON уже может быть легко прочитан людьми. Каждый объект в формате JSON имеет ключ и его значение. В указанном выше примере в роли ключа выступает «hello», а в роли его значения слово «world». Кроме того, в качестве значений в JSON могут быть использованы массивы, числа, литералы (true, false, null), а также строки.

Взаимодействие сервера приложения с сервером баз данных

Взаимодействие сервера приложения с сервером баз данных осуществляется путем выполнения команд–запросов, написанных на JavaScript языке. Это могут быть команды на добавление, модификацию каких–либо значений, либо поиск данных. В разрабатываемом веб–приложении, для взаимодействия баз данных MongoDB и сервера приложения на платформе Node.js был использован модуль Mongoose на стороне сервера приложения. С помощью модуля Mongoose становится возможен удобный доступ к редактированию и поиску информации в базах данных MongoDB.

В ходе разработки веб–приложения были неоднократно использованы команды на создание новых коллекций, добавление и модификацию значений в коллекциях, а также поиск уникальных значений по определенному условию. На рисунке 31 изображен пример добавления новых значений в коллекцию «Olimpiad_CoachLogins», где каждому полю коллекции присваивается уникальное значение, полученное при прохождении регистрации пользователем.

```
var newLogin = new CoachLogin({
    login: cryptoLogin,
    password: cryptoPassword,
    email: cryptoEmail,
    hash: cryptoHash,
    coachFirstName: cryptoFirstName,
    coachSecondName: cryptoSecondName,
    coachCity: cryptoCity,
    coachInstitution: cryptoInstitution,
    coachAvatar: 'default.png'
})
newLogin.save(function (err, newLogin) {});
```

Рисунок 31 – Добавление новых значений в коллекцию

Модификация данных устроена похожим образом, за исключением необходимости провести предварительный поиск необходимых данных перед их модификацией. Пример поиска данных и их последующей модификации изображен на рисунке 32. В данном примере осуществляется поиск конкретного пользователя по его уникальному идентификатору «_id» (строка 459) и если пользователь был найден, осуществляется замена старого значения поля «password» на новое, введенное пользователем. После замены значений производится сохранение изменений в базе при помощи специальной команды «save» (строка 462).

```
459     CoachLogin.findOne({_id: id}, function (err, doc) {
460         if (doc) {
461             doc.password = password;
462             doc.save(function (err) {
463                 if (err) {
464                     console.error('ERROR!');
465                 }
466             })
```

Рисунок 32 – Модификация данных

Поиск данных по базе осуществляется с помощью команд–запросов. Такие команды могут быть как простыми, например нахождение значений по какому–либо параметру, так и сложными, где используются более объемные

конструкции запросов к базе данных. Каждый запрос к базе данных начинается с одной из трех команд:

– find – для нахождения всех значений по указанному параметру;

– findOne – для нахождения первого значения в базе, содержащему указанный параметр;

– findById – для нахождения значения в базе, содержащему указанный уникальный идентификатор «_id».

Пример сложного запроса представлен на рисунке 33. В данном примере осуществлен поиск всех пользователей с логином «user» (строка 230), фамилией «olymp» (строка 231) и возрастом от 17 до 66 лет (строка 232). Полученные данные должны быть выведены в количестве не более десяти штук (строка 233), а также отсортированы по увеличению возраста (строка 234).

```
229         CoachLogin.
230           find({ login: 'user' }).
231           where('name.last').equals('olymp').
232           where('age').gt(17).lt(66).
233           limit(10).
234           sort(age: 1).
235           exec(callback);
```

Рисунок 33 – Сложный запрос к базе данных

2.2.2 Уровень приложения

В разрабатываемом веб–приложении за общий функционал интерфейса, а также связь клиент–стороны и сервер–стороны веб–приложения отвечает фреймворк AngularJS. Он обеспечивает динамическое обновление объектной модели документа (DOM), т.е. программного интерфейса, без его перезагрузки, а также связывает логику клиент–стороны и сервер–стороны веб–приложения. В AngularJS для создания связующего звена между html–разметкой и моделью приложения отвечают специальные

«контроллеры». Задачей контроллера, как и в других MVC фреймворках, является получение данных из модели и вывод их пользователю, либо в обратном направлении, получение данных из представления и отправка их в модель. В свою очередь контроллер может быть связан с сервером приложения посредством http запросов. С помощью http запросов (request) клиент имеет возможность отправлять и получать какие–либо данные (response). На рисунке 34 изображено схематичное взаимодействие всех компонентов системы.

Рисунок 34 – взаимодействие компонентов системы

Модули Node.js

Модули в Node.js – это самописные, либо сторонние библиотеки, предназначенные расширить функционал серверной части разрабатываемого приложения. Применение модулей значительно ускоряет процесс разработки веб–приложения. В разрабатываемом веб–приложении были использованы модули отправки электронной почты, работы с базами данных MongoDB, работы с *.pdf и *.doc документами и другие. Пример подключения модулей изображен на рисунке 35.

```
var express = require('express')
    , app = express()
    , cookieParser = require('cookie-parser')
    , materialDesignIcons = require('material-design-icons')
    , http = require('http')
    , url = require('url')
    , server = http.createServer(app)
    , PDFDocument = require('pdfkit')
    , fs = require('fs')
    , Docxtemplater = require('docxtemplater')
    , mongoose = require('mongoose')
    , bodyParser = require('body-parser')
    , crypto = require('crypto')
    , multer = require('multer')
    , nodemailer = require('nodemailer');
```

Рисунок 35 – Подключение модулей

Реализация паттерна MVC на примере модуля регистрации пользователя

Рассмотрим пример того, как был реализован процесс регистрации в разрабатываемой информационной системе «СФУ Олимпиада». Регистрация проходит в два основных этапа:

– заполнение данных пользователем;

– подтверждение заполненных данных через e–mail верификацию.

Первый этап регистрации состоит из следующих шагов:

– ввод пользователем данных;

– отправка и обработка данных в контроллере приложения;

– отправка данных на сервер приложения;

– повторная проверка данных на сервере, шифровка данных и занесение данных в базу;

– отправка сервером электронного письма с подтверждением регистрации.

Рассмотрим каждый пункт первого этапа регистрации.

Вначале регистрации пользователю необходимо заполнить основные, одинаковые как для тренеров команд, так и для участников данные: логин, e–

mail и пароль. Интерфейс ввода данных для регистрации изображен на рисунке 36.

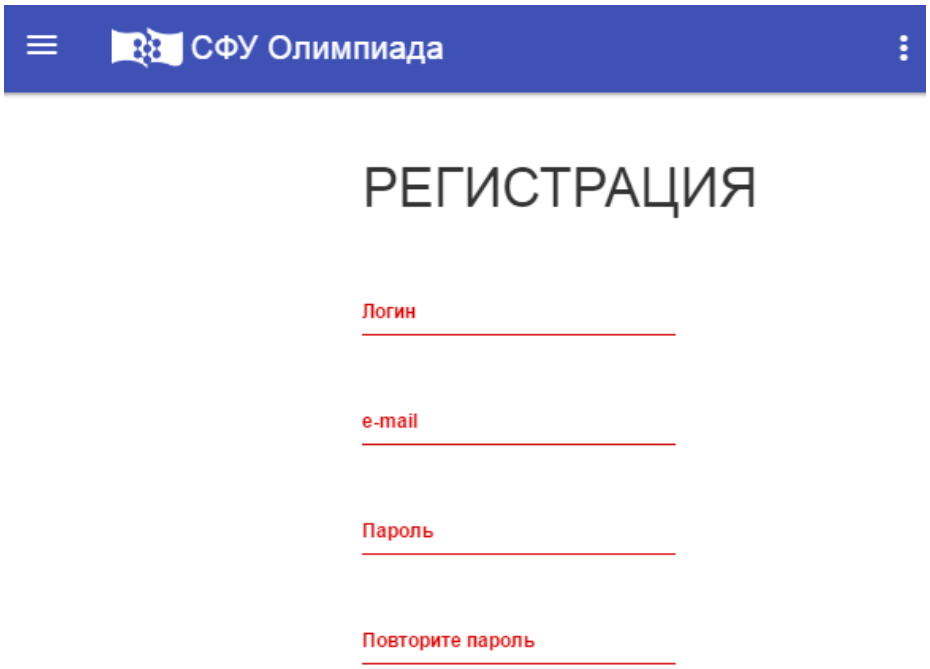

Рисунок 36 – Первичная регистрация пользователя

После введения первичных данных, пользователь выбирает, кем он будет являться: тренером команд или участником. После этого пользователю необходимо ввести личные данные: имя, фамилию, город, образовательное учреждение. Пример заполнения изображен на рисунке 37.

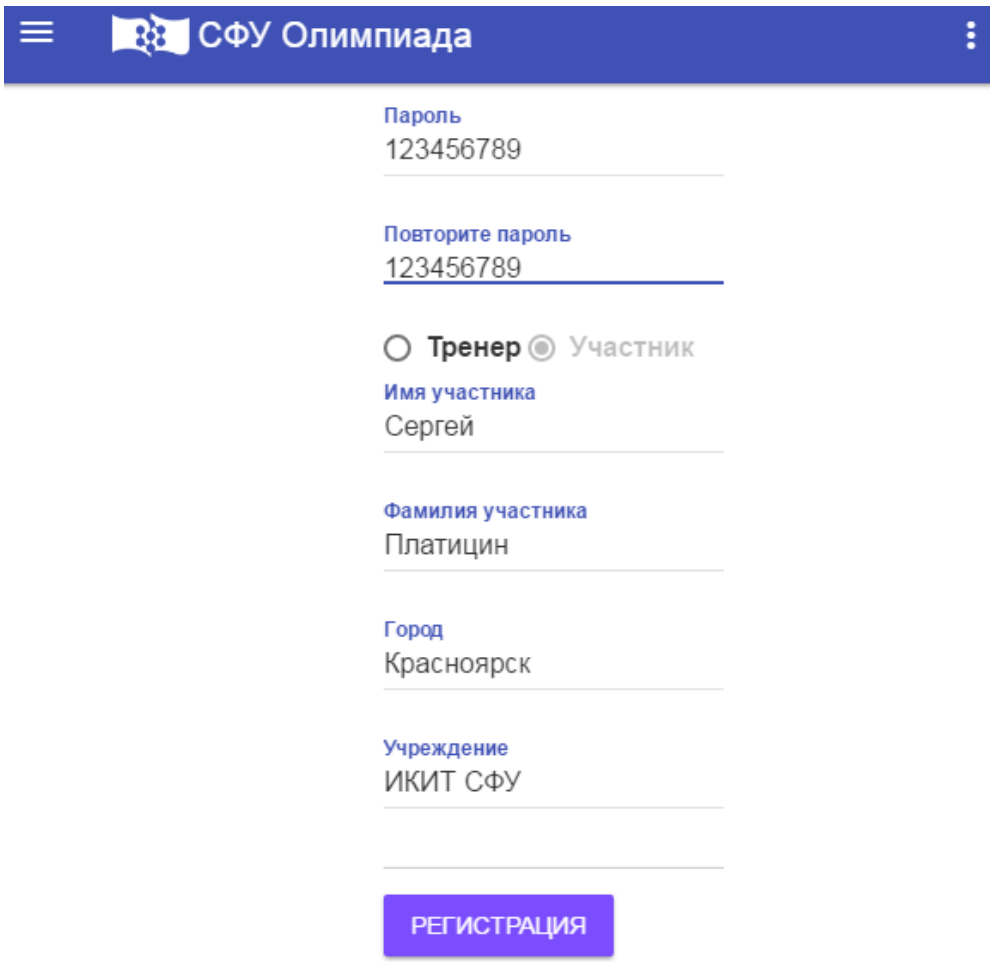

Рисунок 37 – Заполнение полей регистрации

После нажатия на кнопку «Регистрация», система отправляет все полученные данные в контроллер, где с помощью http запроса все заполненные пользователем при регистрации данные отправляются на сервер. Пример кода, отвечающего за отправку данных серверу Node.js, изображен на рисунке 38.

```
$http.post('/sendRegistration', {
    login: $scope.login,
    password: $scope.password1,
    email: $scope.email,
    firstName: $scope.contestantFirstName,
    secondName: $scope.contestantSecondNam
    city: $scope.contestantCity,
    institution: $scope.contestantInstitut
    status: 'contestant'
})
```

Рисунок 38 – Отправка данных на сервер приложения

После получения данных от контроллера, сервером производится шифровка данных. При создании функции шифровки данных был использован модуль для платформы Node.js «Crypto». Пример получения данных сервером и их последующего шифрования изображен на рисунке 39.

```
//Регистрация участников и тренеров
app.post('/sendRegistration', function (req, res) {
    var cryptoLogin = req.body.login;
    var cryptoEmail = req.body.email;
    var bdEmail = req.body.email;
    var cryptoPassword = req.body.password;
    var cryptoHash = "crypto" + cryptoEmail;
    var cryptoFirstName = req.body.firstName;
    var cryptoSecondName = req.body.secondName;
    var cryptoCity = req.body.city;
    var cryptoInstitution = req.body.institution;
    var status = req.body.status;
//Шифрование данных пользователя
    cryptoLogin = cryptThis(cryptoLogin);
    cryptoEmail = cryptThis(cryptoEmail);
    cryptoPassword = cryptThis(cryptoPassword);
    cryptoHash = cryptThis(cryptoHash);
    cryptoFirstName = cryptThis(cryptoFirstName);
    cryptoSecondName = cryptThis(cryptoSecondName);
    cryptoCity = cryptThis(cryptoCity);
    cryptoInstitution = cryptThis(cryptoInstitution);
```

Рисунок 39 – Получение данных сервером и их последующее шифрование

После того, как все данные были успешно получены и зашифрованы, производится проверка переменной «status», отвечающей за тип пользователя, проходящего регистрацию. В зависимости от значения данной переменной («coach» для тренера и «contestant» для участника), система принимает решение, в какую коллекцию должны быть сохранены полученные данные. Пример добавления данных в коллекцию «участники» изображен на рисунке 40.

```
var newLogin = new ContestantLogin({
    login: cryptoLogin,
    password: cryptoPassword,
    email: cryptoEmail,
    hash: cryptoHash,
    contestantFirstName: cryptoFirstName,
    contestantSecondName: cryptoSecondName,
    contestantCity: cryptoCity,
    contestantInstitution: cryptoInstitution,
    contestantAvatar: 'default.png'
})
newLogin.save(function (err, newLogin) {
});
```

Рисунок 40 – Добавления данных в коллекцию «участники»

После того, как все данные были успешно добавлены и сохранены, сервер приложения отправляет электронное письмо с подтверждением регистрации на указанный пользователем e–mail адрес и перенаправляет пользователя на главную страницу. При создании функции отправки электронного письма был использован модуль для платформы Node.js «Nodemailer». Пример отправки электронного письма сервером приложения изображен на рисунке 41.

```
var regMailOptions = {
    from: '"СФУ Олимпиада" <greenfloodmouse@gmail.com>',
    to: bdEmail,
    subject: 'Регистрация ✔',
    text: 'Пожалуйста, подтвердите ваш e-mail адрес',
    html: '<b>Для подтверждения регистрации пройдите'+
    ' по сгенерированной ниже ссылке</b> '+
    '<br> <p><a href="http://localhost:3000/#/activate?hash='
    + cryptoHash + '&password=' + cryptoPassword + '">Ссылка</a></p>'
};
transporter.sendMail(regMailOptions, function (error, info) {
    if (error) {
        return console.log(error);
    }
```

Рисунок 41 – Отправка электронного письма сервером приложения

Пример электронного письма с подтверждением регистрации изображен на рисунке 42.

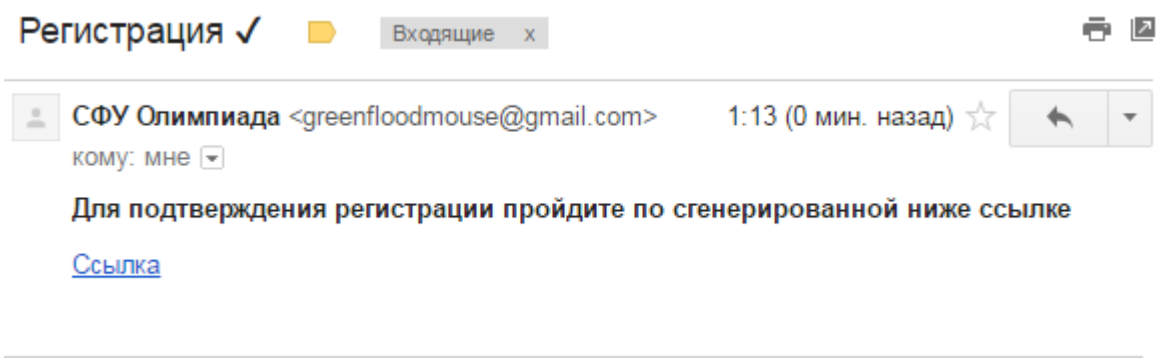

Рисунок 42 – Электронное письмо с подтверждением регистрации в системе

После перехода пользователем по предложенной ссылке, система активирует аккаунт пользователя. Активация аккаунта производится посредством изменения сервером приложения поля «hash» в коллекции «участники» со специально сгенерированного для проверки значения на «isactive». После успешной активации аккаунта, система отправляет сообщение об успехе, и пользователь получает возможность авторизоваться в системе для ее дальнейшего использования. Окно сообщения об успешной активации аккаунта пользователя изображено на рисунке 43.

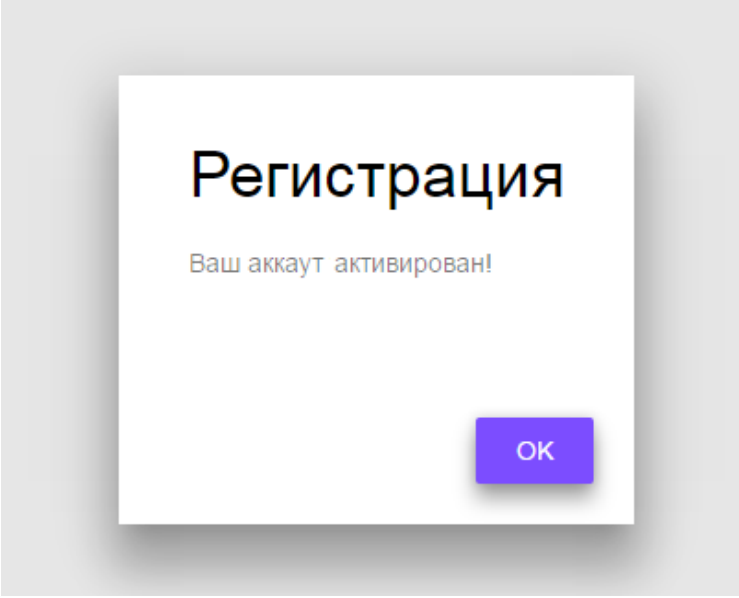

Рисунок 43 – Окно сообщения об успешной активации аккаунта пользователя

ЗАКЛЮЧЕНИЕ

В процессе прохождения выпускной квалификационной работы была спроектированная информационная система в виде веб–приложения, представляющая собой гибкий инструмент работы в сфере олимпиадной деятельности университета.

Веб–приложение «СФУ Олимпиада» позволяет ускорить и сделать более эффективной работу оргкомитета олимпиад Сибирского Федерального университета. Кроме того, архитектура веб–приложения позволяет в короткие сроки разрабатывать и внедрять в систему новый функционал, что дает возможность быстрой автоматизации бизнес–процессов и, как следствие, постоянной поддержки актуальности системы.

В дальнейшем развитии веб–приложения планируется расширение функционала, а также создание отдельного приложения для платформ Android и iOS.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Разработка информационных систем. Ч. 1. Структурные методы : учеб. пособие для студентов направления «Информационные системы» / Ю. А. Маглинец. – Красноярск : Изд–во «Кларетианум», 2004. – 124 с.

2. Информационные системы : учеб. пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 2-е изд. – М.: ФОРУМ: ИНФРА-М, 2014. – 448 с.: ил – (Высшее образование).

3. Разработка и эксплуатация автоматизированных информационных систем : учеб. пособие / Л. Г. Гагарина. – М.: ИД «ФОРУМ»: ИНФРА-М, 2013. – 384 с.: ил – (Профессиональное образование).

4. Проектирование информационных систем : учеб. пособие / Н. Н. Заботина. – М.: ИНФРА-М, 2014. – 331 с. – (Высшее образование: Бакалавриат).